TinyML for Gait Stride Classification

Priyanka Rajendra

TINYML FOR GAIT STRIDE CLASSIFICATION

By

Priyanka Rajendra

Bachelor of Engineering, Electronics and Communication Engineering
Visvesvaraya Technological University, Belgaum, India
2016

A thesis submitted in partial fulfillment of
the requirements for the

Master of Science in Engineering – Electrical Engineering

Department of Electrical and Computer Engineering
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
December 2021

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

December 3, 2021

This thesis prepared by

Priyanka Rajendra

entitled

TinyML for Gait Stride Classification

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Engineering – _Electrical Engineering
Department of Electrical and Computer Engineering

ABSTRACT

Human gait classification and analysis become very important when a person has been diagnosed with a neurological disorder or has suffered an injury which has affected their ability to walk correctly. Gait strides are an important parameter to be studied as it helps the doctor to diagnose any underlying gait condition and evaluate what type of treatment suits the best for the patient’s recovery. Studying gait strides also helps athletes to improve their performance.

In today’s world, machine learning has emerged as one of the most widely used technology for classification and analysis of gait characteristics. TinyML is a field of study in Machine Learning and Embedded Systems that uses the power of machine learning (ML) and runs these models on small, low-powered devices like microcontrollers.

In this study, we delve into the world of TinyML and explore how the gait analysis can be carried out using this technology. The objective of this thesis is accomplished by using multiple sensors to collect gait stride patterns through a series of trials. We will be using STM’s SensorTile for collecting data. Three different types of stride motions are recorded for classification of short, long, and normal strides. We will use various machine learning/deep learning models to train the data and classify the stride lengths. Next, we’ll improve the model performance by using methods such as feature selection and tuning of various model parameters. On obtaining an optimized model, the model is translated to an embedded TinyML model for real-time evaluation. The classification and regression TinyML model are evaluations with ground truth values observed using a depth sense camera with skeleton tracking algorithms.

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Dr. Venkatesan Muthukumar, who has always supported me and pushed me to do my best throughout the course of my Master’s degree. I would like to thank him for his guidance and for sharing his invaluable knowledge with me for this thesis study. I am also grateful to him for helping me to get all the required components for my thesis study.

I would like to thank my thesis committee members Dr. Mei Yang, Dr. Emma Regentova, Dr. Si Jung Kim and Dr. Juyeon Jo for their support and feedback. I am grateful for having them as my committee members.

I would also like to thank Sreeja Kanchi, Ashika Mettu and Kavitha Krovi for helping me to collect the data required for my study.

I am thankful to the Department of Electrical and Computer Engineering and to our department chair Dr. Biswajit Das for providing me Teaching Assistantship and a great learning environment to pursue my MS degree.

Last, but not the least, I would like to extend my heartfelt gratitude to my family and friends, especially my parents Rajendra Kondebettu and Lata Rajendra, my in-laws Dr. Anantha S. and Usha C.K., and my husband Malatesha S.A., who have always stood by me in the most difficult times and always motivated me to achieve the best in life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Walking is an everyday activity, to which we do not give much thought. Walking in a correct manner is very essential to prevent any spine or leg issues. Each person's walking pattern or gait is unique. Gait is defined as a way or style in which a person walks. A gait cycle is a repetitive pattern which involves steps and strides [1].

Some people may have abnormal gait patterns, which they might have been following for years, leading to larger health problems in the future. Also, when someone has met with an accident, which has caused injury to their leg, their normal gait can get altered. An individual suffering from certain neurological illness such as cerebral palsy also show abnormal gait patterns. Many children learning to walk may develop a habit of toe walking, wherein the child walks on the balls of his/her feet. If a child has not outgrown this habit, it might be a sign of underlying medical condition. All these conditions require gait analysis, so that the abnormal patterns can be evaluated, and the necessary treatments can be given to them.

Gait analysis is the process of analyzing and characterizing the gait parameters [2]. There are various ways in which the gait analysis can be carried out. A person may be asked to walk in a certain way and the walking patterns may be recorded. Stride analysis is one of the common ways to describe a gait pattern. The step length and stride length are the important parameters to be recorded. Step length is the length measured parallel to the Line of Progression of the body, from the heel of the previous footfall to the heel of the current opposing footfall [3]. Stride length is the distance measured parallel to the Line of Progression, between the Posterior Heel points of two consecutive footprints of the same foot [3]. Figure 1, shows the various gait parameters such as step length, stride length, step angle, step width and gait cycle. Stride length and stride angles give a lot of information about a person's walking pattern. Thus, they become important parameters in gait analysis.

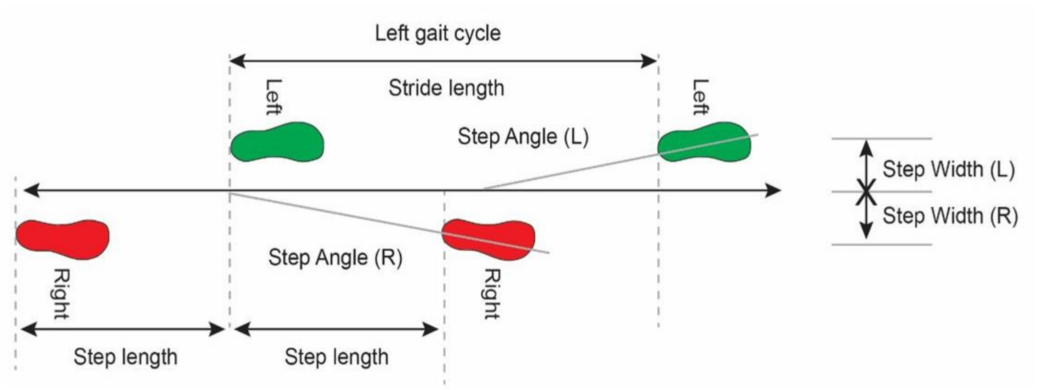


Figure 1: Gait parameters

There are three different ways in which gait analysis or recognition can be carried out- Machine Vision Based (MV), Floor Sensor based (FS) and Wearable Sensor based (WS) [4]. The first approach consists of using various cameras, either digital or analog, for capturing the gait data. The Floor Sensor based method, the sensors are placed along the floor, mainly a mat or any equipment like a treadmill. The Wearable Sensor based approach of gait analysis consists of attaching the sensors on the body of the person; in the case of gait analysis, it's mainly attached to the ankle or knee or placed in the shoes.

Machine Learning is an efficient way to carry out the classification of the various gait patterns. There are lot of benefits of machine learning techniques as compared to conventional techniques. Some techniques, especially, deep learning techniques can determine a wide variety of complex classification by using powerful algorithms and can also be used for a large dataset. Combining the machine learning with embedded systems and IoT, useful systems can be designed which can be especially helpful in healthcare, in our case the gait analysis. Use of TinyML is one such solution.

TinyML is a field of study which combines two domains- Machine Learning and Embedded system, which run on small, low powered devices such as a microcontroller. TinyML is an emerging technology which has its uses in many areas such as healthcare, industries, agriculture, etc.

1.2 Problem

The main question we need to consider is why there is a need for TinyML when there are several IoT devices which have a similar usage. There is a need for incorporating machine learning into embedded devices as the IoT devices have several shortcomings.

Most IoT devices consume less bandwidth, but as the number of devices grow over the network, they all end up taking up large bandwidth. In such cases, once has to ensure if their internet is powerful enough to accommodate all these devices and not cause interruption of the system.

Latency is another issue with IoT devices. As most IoT devices send the out to the cloud to be processed, there is some time lag for the data to be processed and a response to be sent back. The network speed also matters in this system. In cases, where the internet speed is slow, the response is delayed. So, a better solution is to use the machine learning algorithms on the edge devices itself.

Another problem in IoT systems is the large usage of energy. Using GPUs to run the algorithms require a lot of power. For example, in May 2020 an algorithm named GPT-3 was released. This algorithm has a network architecture consisting of 175 billion neurons and uses approximately 3 GWh of electricity [5]. TinyML systems are much more energy efficient as most of these units, which consist of microcontrollers, can operate on batteries.

The IoT devices mainly collect the required data and send it over the cloud for the further processing. Since this data is transmitted over a network, it can be open to many vulnerabilities, thus causing an issue to the privacy and security of the data. This data can be hacked before it reaches the destination, which may not only cause loss of money to companies providing the service, but it can also be life threatening in some cases. The data collected by IBM Cost of Data Breach report in 2020 showed that the average cost of a data breach was USD 3.86 million [5].

1.3 Impact

TinyML devices are much more secure, much more energy efficient, have a reduced latency and have a low cost. Due to this reason, they have a lot of uses in a wide variety of industries.

Wearable technology has been used for various purposes such as to monitor one’s fitness levels, tracking health parameters, usage in sports activities, etc. Using wearable sensors with the TinyML system can make an efficient system for collecting data and getting the results instantly. The various operations can run locally on devices without having to worry about the network connections being slow or the data being leaked. It also promotes better user experience as latency can completely change user experience when using the wearables.

Recent advancements have increased the computational power of microcontrollers. Deploying deep learning models on microcontrollers is also beneficial as these devices use less energy compared to the systems with powerful GPUs that require a lot of processing power. Thus, TinyML for gait analysis can be an efficient method to learn about a person’s gait patterns and stride measurement.

CHAPTER 2

LITERATURE SURVEY

2.1 Prior Solutions

W. Niswander and K. Kontson [6] recorded the gait data using IMUs by asking seven participants to complete a straight walking task and an obstacle navigation task. They made use of five different algorithms to identify the toe-off and heel strike gait events. They found three factors affecting the gait measurements- the algorithms used, the location on the body where the sensors are placed, and the type of walking task undertaken.

C. Monoli et al. [7], proposed an underwater IMU system similar to the one's used on land. They carried out three different trials. In the first trial they carried out a three-way performance analysis between the IMU, optoelectronic and motion capture systems. This setting was carried out on land. In the second trial, a motion-tracking camera is used along with the IMU, and a comparison is carried out between the land and underwater data. They concluded that the wearable IMUs can be effectively used both on land and underwater.

K.P. Rustia and S. Hosseinpouli [8] carried out the gait classification using two sensors and a BeagleBone IoT platform. They carried out three classifications- long stride, short stride and sitting. They used the sensors on the same leg- one on the ankle and the other on the knee. There were two main attempts, with 10 trials of data collected for the former and 15 trials of data for the latter. In this experiment they were successfully able to classify between the short and long strides and sitting.

Zexia He et al. [9] used a motion sensor and six pressure sensitive electric conductive rubber sensors and designed a system for the elderly with knee osteoarthritis to calculate the knee adduction movement (KAM). The system determined the variation in KAM before and after gait training.

Along with wearable devices, smartphones and sensing fabrics are also used for monitoring a person’s activity and record the required data. Smartphones have various in-built sensors which can be useful when used with apps designed to monitor data such as walking distance, motion type, speed, number of steps covered, etc. Smart fabrics may be either electronic components attached to the fabric, or it might be a special type of fabric with electrically conductive filaments [10].

2.2 Our Solution

We have proposed a model to collect data from sensors and classify between three types of strides, namely- short stride, long stride and normal stride. We have used a depth sense camera with skeleton tracking algorithms to obtain the ground truth and get the coordinates of the full body in 3D.

The sensors used are STMicroelectronics Sensortile, which are attached to both the left and right ankles. A Raspberry pi 4 is used to collect the data and we have used Intel RealSense

D435i depth sense camera with the skeleton tracking SDK from Cubemos. The data collected from sensors are accelerometer and gyroscope data. Accelerometers measure linear acceleration and 3-axis gyroscopes measure angular rate.

The data collected is trained using deep learning. The model is improved by tuning various machine learning parameters. The model is then converted into a model compatible with the embedded device and loaded into it, which is then able to classify the data. We make use of libraries such as Tensorflow and Keras to build the machine learning model. The Tensorflow model is converted into a Tensorflow Lite model, which helps to run this model on embedded and mobile devices.

CHAPTER 3

MACHINE LEARNING ON EMBEDDED DEVICES

3.1 Machine Learning

Over the previous few years, many powerful analytical models have been created using machine learning, that have changed the world and eased out so many processes which required human intervention. Deep learning, which is a class of machine learning, carries out the function almost mimicking a human brain. Recent advancements in embedded systems have also made it possible to run complex machine learning models on it, which has changed the way sensors and wearables can be used.

Machine Learning techniques can be mainly classified into 3 groups: Supervised, Unsupervised and Reinforcement learning. Each type is used for various purposes depending on the data and outcome to be achieved. There are two types of data used when carrying out machine learning- labeled and unlabeled data. In labeled data, the input features and output are all named or labeled. This type of data requires human intervention to carry out the labelling process. Unlabeled data, on the other hand, requires very minimal or no human intervention as either very few parameters or none of them are labelled.

Supervised learning uses labeled data to carry out the training and predict the outcome. This technique is much simpler as compared to the other two techniques. Supervised learning is mainly used to carry out classification or regression problems. In classification problem, the data is divided or grouped into specific categories. Linear classifiers, Support Vector Machines (SVMs) and random forest are some examples of classification algorithms [11]. In regression method, the correlations are calculated between dependent and independent variables. Linear regression and logistic regression are the most common types of regression algorithms used.

Unsupervised learning is carried out on unlabeled data. They are used to analyze and cluster the unlabeled data [11]. Unsupervised techniques group data based on similar patterns. k-means clustering is one of the most common unsupervised learning examples.

Reinforcement machine learning is mainly learning by trial and error. The model learns from its experiences and feedback, very much like how human beings would learn. Reinforcement learning uses rewards and punishments for it's correct and incorrect behavior. Unsupervised learning and Reinforcement learning may seem similar to each other, but their goals are different from each other. Unsupervised learning involves finding similar pattern sin the dataset whereas reinforcement learning aims at maximizing its total reward. The designer is responsible to set the reward policy or the set of rules by which the model scores the reward. Markov Decision Processes (MDPs) and Q-learning are some of the examples of reinforcement learning [12].

Deep learning is a branch of machine learning, which mimics the human brain. It consists of neural networks or neurons of three or more layers. The neural network basically has an input layer, an output layer and hidden layer(s). Perceptron was the first artificial neuron developed by the scientist Frank Rosenblatt. Over the years, deep learning has become a powerful tool because of its ability to handle large amounts of data. Some of the complex algorithms in deep learning has led to handling a lot of complex problems. Some of the most popular deep learning algorithms are CNN, LSTM and RNN.

Convolutional Neural Networks or CNN in short are one of the most popular deep learning models used. CNNs were first developed in the 1980s. The winner of the 2012 ImageNet Computer Vision contest was the AI system AlexNet which was developed by creator Alex Krizhevsky [13]. This system used CNNs which roughly imitates the human vision. This gave a turning point for CNNs or deep learning in general in the field of machine learning. CNNs uses the mathematical function of convolution that uses two functions and produces a third function based on how the shape of one is modified by the other.

RNN or Recurrent Neural Networks are type of neural networks that use the previous outputs as inputs, while having hidden states. They use sequential data or time series data. They find their usage mainly in Natural language processing (NLP), image captioning and speech recognition. One of the most common examples are Apple's Siri and Google's voice search [14].

Another popular deep learning model is LSTM model. Long Short-Term Memory (LSTM) networks are a type of recurrent neural network used in sequence prediction problems which are capable of learning order dependence. A typical LSTM network consists of different memory blocks called cells. The two important states are the cell state and the hidden state. The memory blocks are responsible for memorizing things. The manipulation to this memory is done through three types of gates- Forgot Gate, Input Gate, and Output Gate.

There are a lot of advantages of using machine learning. Machine learning helps to automate the process or work. This helps to reduce human intervention, especially in repetitive tasks. Machine learning helps in dealing with large amount of data which is difficult to be carried out by using simple applications and manual work. They also help in identifying hidden patterns, which is might not be easily identified by human eyes. This is helpful for providing suggestions or recommendations to users when using e-commerce sites like Amazon and entertainment sites like Netflix. Machine learning also has its use in wide variety of applications such as business, healthcare, e-commerce, educational purposes, research, etc.

As with most things, machine learning also has its fair share of drawbacks. Machine learning requires a lot of resources, and it can also be time consuming. Some algorithms especially deep learning algorithms require large amount of processing power, and it may also take days to complete the learning process. The accuracy and precision obtained by the model is also dependent on the dataset used. It requires a huge amount of unbiased data and preprocessing of this data is essential to obtain good results. Developing complex algorithms and running it may also prove to be expensive and requires a lot of investment.

3.2 Embedded systems

An embedded system is a combination of hardware and software which is designed to serve a specific function. Embedded systems are like miniature versions of computer hardware with software embedded on them. These systems may either work independently or they can be a part of a larger system.

Embedded systems consist of the following basic components [15]:

- Power Supply- This is essential to turn on the embedded device. Power supply can either be a battery or a power adaptor. Smooth and Continuous supply of power is essential for the embedded device to function properly.
- Processor – Just like any computer, embedded devices also consist of a processor which acts as the brain of the system. There are various processors such as 8-bit, 16-bit, 32-bit, etc. For large operations, higher bit processors are required to process the data efficiently.
- Memory – There are two types of memory uses- Random Access Memory (RAM) and Read-only Memory (ROM). RAM is a volatile memory, i.e., the data can be only stored on a temporary basis. The data is generally lost after the system is turned off. ROM is a non-volatile memory which is basically used to store the system program. When the system is turned on, it fetches the boot firmware from the ROM.
- Timers- Timers in the embedded system help to keep a track of time for various events occurring in the system.
- Communication Ports- These ports help to establish connection with other systems. There might be multiple communication ports like serial ports, UART, USB, Ethernet, Bluetooth, etc.
- I/O ports – I/O or Input-Output ports are used to receive and send the data respectively. Input may be given by the user, other systems or by using various sensors. The output can either be data sent to another system or it might be an LED display shown on the embedded system.

- Additional hardware – Since every embedded system have different purpose or function, additional hardware may be placed according to the needs. These may be sensors, additional timers/counters, LED display, etc.

There are various advantages and disadvantages of embedded systems. They are portable and use less power. They can be easily customized according to user's needs. Most of the embedded devices are cheap and affordable to common man. Embedded systems require high effort for development and some systems may also take up a lot of time for troubleshooting the issues. Also, once the embedded device is configured, it will be difficult to reverse it to its original form. They also have limited memory which can be an issue if the system wants to store large amount of data.

There are wide range of applications of embedded systems due to its diversity in functions. They are used in automobiles especially for applications such as cruise control, navigation systems, sensor functionality for various purposes such as airbags and collision mitigation systems. The mobile phones we use also consist of embedded systems. Embedded systems also find a lot of usage in various industries mainly for monitoring and control purposes. These devices are also very useful in healthcare. Embedded devices are used to track a patient's health condition in the form of wearables or by being a part of various medical equipment.

3.3 Machine Learning on Embedded Devices and Literature Review

Embedded Machine Learning, also known as TinyML is a field of study combing both machine learning and embedded system, creating tiny yet powerful systems to carry out many repetitive tasks. As discussed in the previous Chapter 1, using the IoT systems are not always feasible due to various issues such as connectivity issues, privacy issues and latency. To reduce these problems, the solution would be to at least process some or most of the data locally using embedded devices. Embedded Machine learning is a part of edge computing, which helps to process the data locally rather than on cloud.

S. Branco et al. [16] highlighted the drawbacks of cloud-dependent system and provided insights on how to implement standalone Machine Learning models into MCUs with resource scarcity and field-programmable gate arrays (FPGAs) that are typically at the end of any network. They also provided various guidelines, tools and mentioned about the various optimization techniques for implementing ML models in resource scarce MCUs.

T.S. Ajani et al. [17] conducted a survey on how Machine learning is used within embedded and mobile devices. They explored models such as hidden Markov, k-nearest neighbors, support vector machines, Gaussian mixture models and deep neural networks. They also delved into various optimization techniques which are used in recent times to incorporate such memory-intensive algorithms into the embedded devices.

C. Banbury et al. [18] studied about the neural architecture search (NAS) to address certain challenges of tinyML such as limited MCU memory, latency, and energy constraints. They have designed the MicroNet models and deployed them on MCUs using Tensorflow Lite

Micro. These performed exceptionally well on tasks such as visual wake words, anomaly detection, and spotting of audio keywords.

M. Arabi and V. Schwarz [19] addressed the various constraints in embedded machine learning such as power, cost, privacy and performance, and proposed several optimization methods to overcome the challenges. They suggested software optimizations consisting of architectural, algorithmic and memory-usage optimizations, and hardware optimizations to improve the performance of these systems.

CHAPTER 4

EXPERIMENTAL SETUP AND DATA COLLECTION PROTOCOLS

4.1 Hardware used for the experiment

4.1.1 ST Microelectronics SensorTile

The sensors used for this experiment are ST Microelectronics SensorTile IoT module. The zoomed image of the STLCS01V1 SensorTile component board is shown in Figure 2.

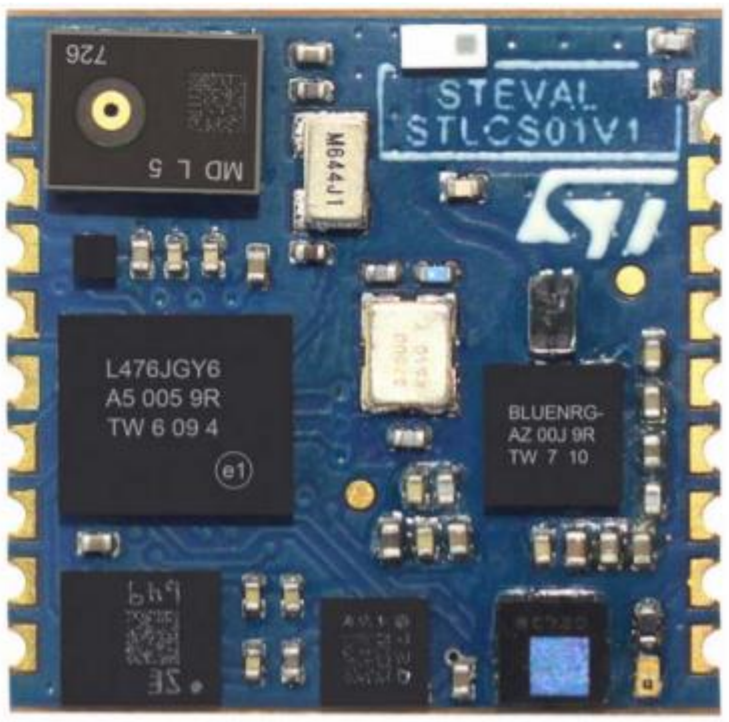


Figure 2. STLCS01V1 SensorTile component board

The main components of the SensorTile module are as follows [20]:

- STM32L476JG – 32-bit ultra-low-power MCU with Cortex®M4F
- LSM6DSM – iNEMO inertial module: 3D accelerometer and 3D gyroscope
- LSM303AGR – Ultra-compact high-performance eCompass module: ultra-low power 3D accelerometer and 3D magnetometer
- LPS22HB – MEMS nano pressure sensor: 260-1260 hPa absolute digital output barometer
- MP34DT05-A – 64 dB SNR digital MEMS microphone
- BlueNRG-MS – Bluetooth low energy network processor
- BALF-NRG-02D3 – 50 Ω balun with integrated harmonics filter
- LD39115J18R – 150 mA low quiescent current low noise LDO 1.8 V

Some additional features include 2V to 5.5V power supply, external interfaces such as UART, SPI, SAI, etc., can be plugged or soldered and can be connected to mobile phones using STBLESensor app. Also. SensorTile also support software libraries such as SensorTile firmware package that helps in raw streaming of data via USB and data logging using SDCard- STSW-STLKT01 and BlueST-SDK for Android, IoS and Python platforms, which helps to access data sent by a device using Bluetooth Low Energy (BLE) implementing the BlueST protocol.

SensorTile makes use of Bluetooth Low Energy (BLE) instead of the regular Bluetooth used by our computer and mobile devices. The main difference between the regular Bluetooth and Bluetooth Low Energy is the low energy consumption by the latter as compared to the former. This helps to transmit data from a small device, running on a small battery, for a longer period of time. BLE operates in the 2.4 GHz band and remains in sleep mode unless a connection is initiated. BLE also has a faster connection speed as compared to the Bluetooth. It requires only few milliseconds to establish a connection. BLE devices can also connect to larger number of other devices.

The SensorTile development kit also comes with a cradle, which we are using for the experiment, onto which the SensorTile chip can be soldered. This cradle provides features such as battery charger, humidity and temperature sensor, SD memory card slot, USB port and breakaway SWD connector [20]. Figure 3, shows the top and bottom view of the cradle or the official name STLCR01V1 SensorTile component board.

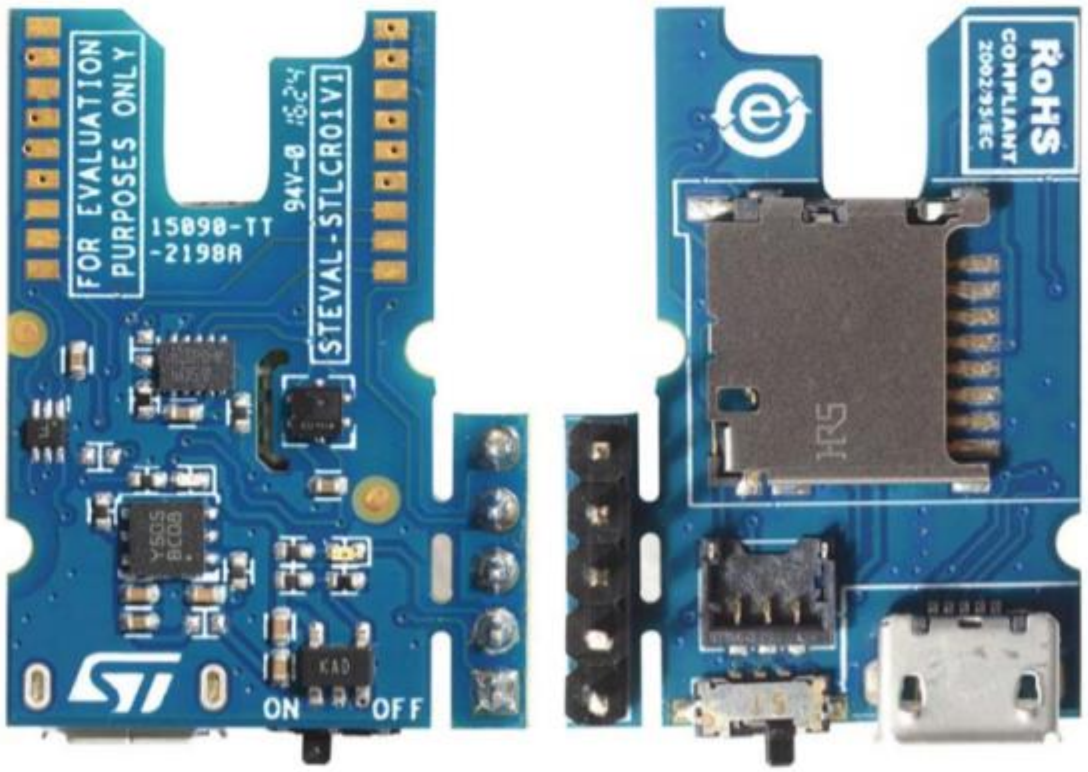


Figure 3. STLCR01V1 SensorTile component board

The SensorTile cradle board has the following features [20]:

- Sensortile Cradle board with SensorTile footprint (solderable)
- STBC08PMR – 800 mA standalone linear Li-Ion battery charger
- HTS221 – capacitive digital sensor for relative humidity and temperature
- LDK120M-R – 200 mA low quiescent current very low noise LDO
- STC3115 – Gas gauge IC
- USBLC6-2P6 – very low capacitance ESD protection
- USB type A to Mini-B USB connector for power supply and communication
- microSD card socket
- SWD connector for programming and debugging

The SensorTile IoT module is soldered onto the cradle and then placed into a small plastic case which helps to use the device as a wearable. Figure 4, shows the SensorTile system placed in the plastic case.

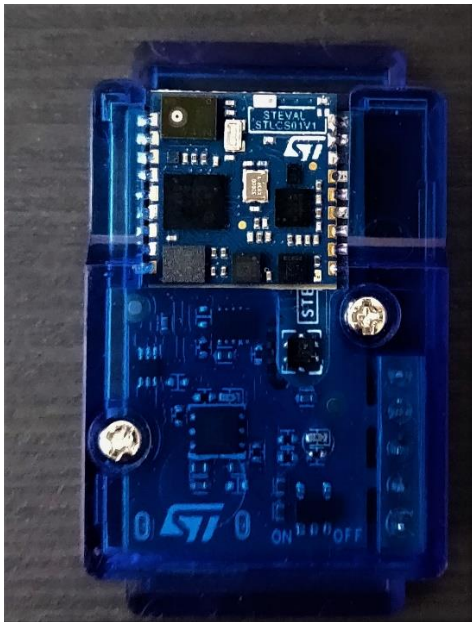


Figure 4. SensorTile Cradle placed in plastic case.

4.1.2 Raspberry Pi 4

Raspberry Pi 4 is the latest model in the raspberry pi series. It has a Broadcom BCM2711 64-bit quad-core processor, dual-display support at resolutions up to 4K using the micro-HDMI ports, hardware video decode at up to 4Kp60, supports upto 8GB of RAM, Bluetooth 5.0, Gigabit Ethernet, dual-band 2.4/5.0 GHz wireless LAN, and USB 3.0 [21]. Figure 5, shows the Raspberry pi 4 Model B device.

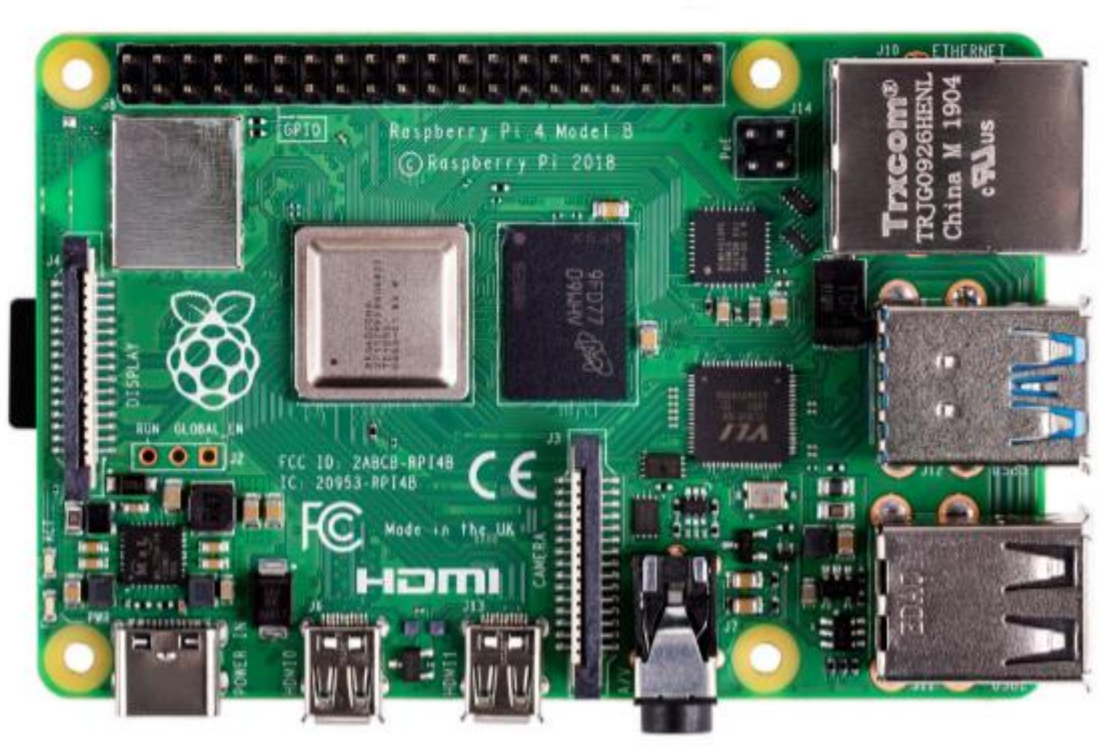


Figure 5. Raspberry Pi 4 Model B

Raspberry Pi 4 can also connect to standard Bluetooth as well as BLE devices. It supports Micro SD card for flashing the operating system and to store data. Raspberry Pi 4 is powered using a USB Type-C cable supplying at least 5V 3A of power. This device can be easily connected to a laptop using Wi-Fi or by using an Ethernet cable.

4.1.3 Intel RealSense Depth Camera D435i

The standard digital cameras output 2D images. The pixel values are associated with the RGB values. A picture/photo is made up of millions of such pixels. A depth camera has pixels with different numerical values. These numerical values are the distances from the camera, known as depth. Some cameras have only the depth camera, whereas some have both depth and RGB camera. The Intel RealSense depth camera D435i comes in the latter category, consisting of both depth camera and RGB camera. It also has an addition of an inertial measurement unit (IMU) along with the depth sensing capabilities.

There are different types of depth sense cameras. The Intel RealSense 400 series are stereo depth cameras [22]. These cameras can use any light to measure depth. They have two sensors, and the stereo cameras use the images from these two sensors for comparison. This gives the depth information similar to how our eyes perceive the depth.

As mentioned previously, the D435i has an IMU. The IMU consists of numerous sensors with gyroscopes which help in detecting movement and rotation in three axes. They also detect the pitch, yaw and roll. Pitch, yaw and roll are the rotations which takes place around the side-side axis, vertical axis, front-back axis respectively.

Figure 6, shows the Intel RealSense depth camera D435i. The right and the left imager help in capturing images. The IR projector emits infrared light to improve the accuracy of the data. The RGB module helps in displaying the RGB image. These cameras find a great use in virtual reality and robotics.

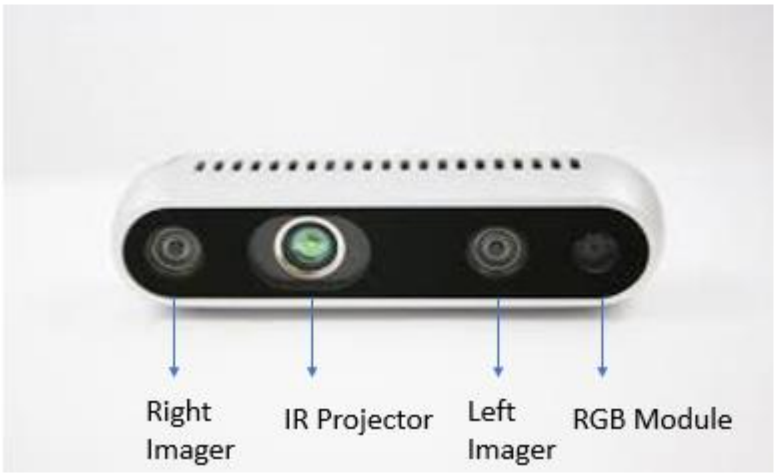


Figure 6. Intel RealSense depth camera D435i

4.2 Software and Libraries used for the experiment

4.2.1 Python

The coding for this experiment is carried out using Python language. Python is an interpreted programming language, which was created by Guido van Rossum, and it was released in 1991. Python is a powerful programming language which can be used on multiple platforms and for multiple applications. It's ease of use and simple syntax has made it popular especially in the field of machine learning. Python supports procedural, object-oriented and functional programming.

There are various Interactive Development Environments (IDEs) and text editors used for Python programming. For this experiment, we have made use of Jupyter Notebook IDE. Jupyter Notebook is an open-source web-based IDE which is used in various applications such as machine learning, statistical modelling, data visualization, etc. This application helps in line-by-line execution of code. This makes it easier to debug issues encountered in the code.

One of the main reasons why Python is useful in machine learning is the availability of numerous libraries, which can be imported and used in our code. This saves a lot of time and effort while building machine learning codes. Among the many libraries, Keras is one of the most important libraries which we are using in our Python code. Keras is an open-source library used in Python, when dealing with neural networks. Keras provide high-level APIs and is very helpful when creating deep learning models. Keras serves as a wrapper to TensorFlow framework and makes it easier for building and training deep learning models.

4.2.2 TensorFlow and TensorFlow Lite

Another important library used in building our machine learning model is TensorFlow. TensorFlow is an open-source library used for machine learning and deep learning developed by Google Brain. TensorFlow makes it easier for developers to create and train ML models. It also provides APIs in Python, C++ and also for some other programming languages.

The Machine Learning model created is not compatible with embedded devices such as microcontrollers due to their limited memory. To solve this issue, Google developed the TensorFlow lite library. TensorFlow Lite helps in running the ML models efficiently on embedded devices and mobile phones. To be able to do one has to deal with some trade-offs such as no support on training, just running inference on models that were previously trained on another device or cloud, non-availability of certain data types such as double, and also absence of some less used functionalities [23]. These trade-offs make it possible to reduce the size of the model and allow it to be added into devices with limited memory space.

4.2.3 Skeleton Tracking SDK by Cubemos

Skeleton Tracking SDK by Cubemos is a software package that provides a deep learning based 2D or 3D tracking of the full body for embedded and other hardware. This software development kit helps in tracking 18 joints simultaneously and can also track unlimited people who are captured in the scene. They also provide the coordinates of the joints, thus helps a person to measure the distance between various joints.

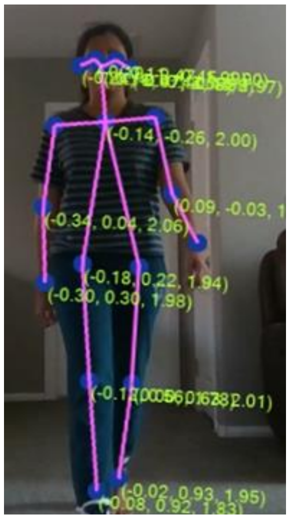


Figure 7. Skeleton Tracking SDK used with the D415 depth camera

Skeleton Tracking SDK is compatible with C, C++, C# and Python for both Windows and Linux platforms. It can also be used with the Unity software which is mainly used to design games and Virtual reality (VR). It is also compatible to use with the Intel RealSense depth cameras- D415 and D435. These depth cameras can be used as an input and the SDK provides the pre-built binaries for the librealsense-2.36 SDK from Intel RealSense. Figure 7, shows an example of the Skeleton Tracking SDK used with the D415 depth camera.

4.2.4 Additional Software Tools and Applications

Along with the main software mentioned above, certain other software was used for this experiment which were useful for the whole process. The first is the Raspbian, which is an operating system used for Raspberry Pi hardware. This is the primary OS used with Raspberry pi and for the desktop it uses a modified Lightweight X11 Desktop Environment (LXDE) with the Openbox stacking window manager.

The next set of tools are the VNC (Virtual Network Computing) Server and VNC Viewer. VNC is a graphical desktop-sharing system which allows the user to establish a remote connection with another computer and control it from your device/system. In our experiment, this server is used to control the Raspberry Pi 4 through the laptop. VNC Viewer is another software that helps to view the desktop, or the contents of the other system which is being controlled by your device. It helps to contact the server and edit the contents it receives.

Another library that plays an important role to work with the SensorTile data is the BlueST SDK. BlueST SDK is a library that helps in accessing and processing the data exported by the Bluetooth Low Energy (BLE) devices which use the BlueST Protocol. This SDK works on Android, iOS and Linux platforms.

4.3 Setup and Procedure

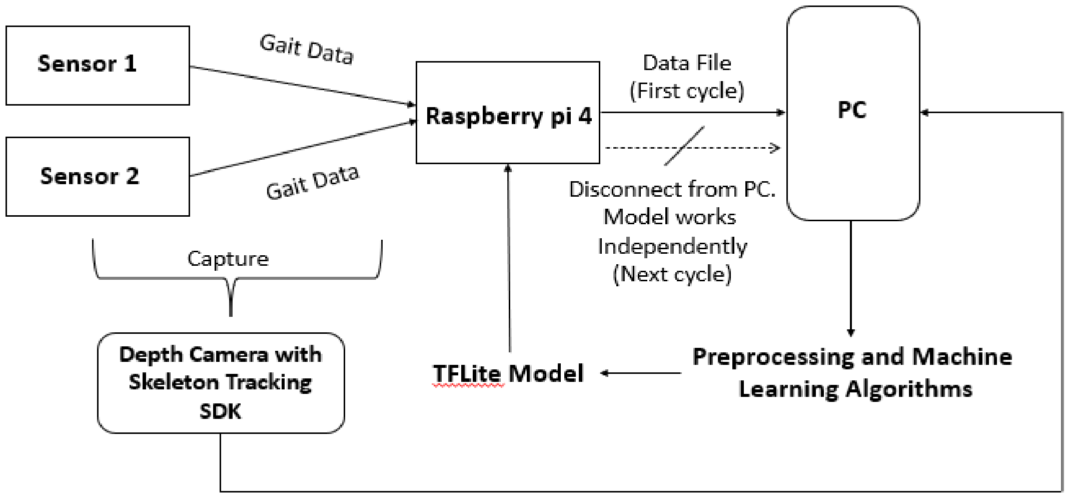

Sensor 1
Sensor 2
Gait Data
Gait Data
Raspberry pi 4
Data File (First cycle)
PC
Disconnect from PC. Model works Independently (Next cycle)
Capture
Depth Camera with Skeleton Tracking SDK
TFLite Model
Preprocessing and Machine Learning Algorithms

Figure 8. Workflow of the Gait classification system

Figure 8-shows the workflow of the system, the various components used and the data flow. For this experiment, we are using two SensorTile devices from STMicroelectronics. The names of the two devices are-AM1V330 and AM1V410. The AM1V330 is placed on the right ankle and the AM1V410 is placed on the left ankle. Figure 9, shows the sensors tied to both the ankles and the orientation of these devices. We are using the accelerometer and gyroscope data from these devices.

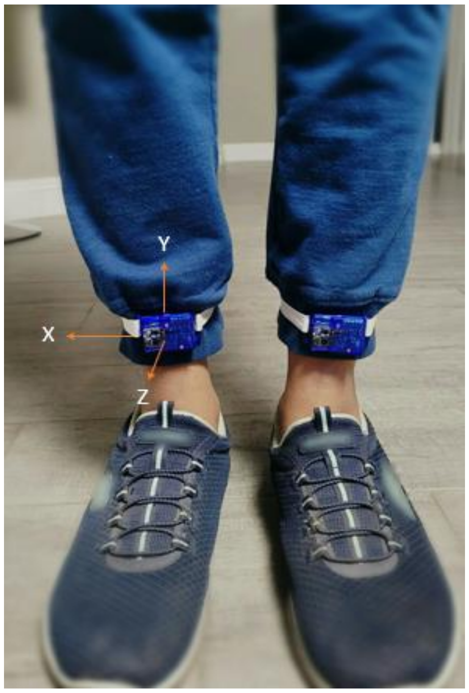


Figure 9. SensorTile devices placed on the right and left ankles

The setup for collecting the data is shown in Figure 10. The total length of the walking path marked is 14 ft one side. If we consider walking in both directions, then the total length will be 28 ft. Each segment marked is of length 1.5 ft.

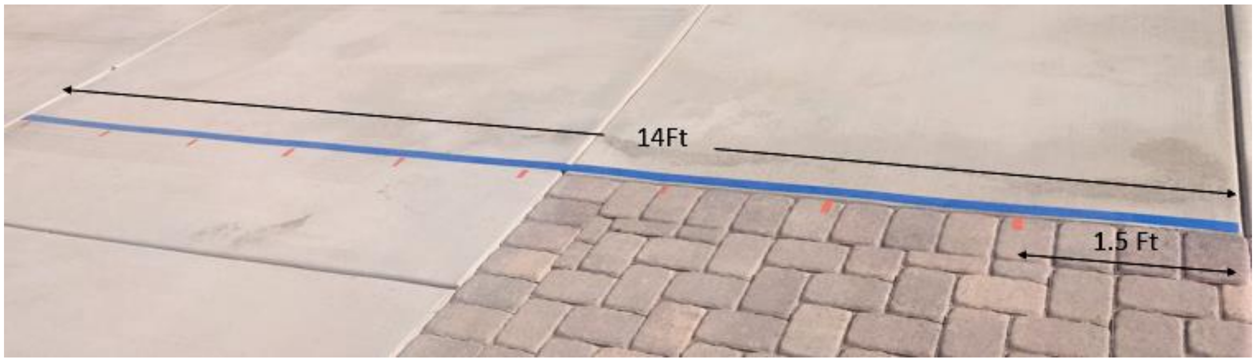


Figure 10. Walking trail for collecting gait data

Three different types of gait data are collected-short stride, normal stride and long stride. For short strides, both the feet are placed within each segment; for normal stride the heel of the second foot is placed on the marking of each segment; and for long stride, the second foot is stretched as much as possible, without straining too much, and placed as far as possible within the marking of each alternate segments.

We have collected data from 5 different people with heights ranging from 5ft to 6ft. A total of 5 trials for each type of stride from each person is carried out, i.e, a total of 15 trials for each person. The sensor devices are both paired with the Raspberry Pi 4 via BLE, and the data is recorded on the Raspberry Pi 4. The data is later sent to the laptop, which is connected to the Raspberry Pi 4, converted to a CSV file, preprocessed, and trained using various ML algorithms to obtain the best results.

The strides are also captured with the Intel RealSense depth camera D435i with the skeleton tracking SDK integrated to it. This gives the full body tracking along with the coordinates of the joints. A front view and a side view of the motion is captured.

The trained ML model and then converted to a TFLite model and deployed to the raspberry pi 4 along with a text file with the labels. We require inferences on this device to execute the model and make predictions with the input data provided. Basically, the main steps in executing the TFLite model on embedded and edge devices involve loading the TFlite model along with the label file, transforming the supplied input into the format which matches with the input supported by the model, running inferences such as allocating tensors, using interpreters, and invoking them, and finally using the data from the inferences to provide an output which the end-user desires.

CHAPTER 4

RESULTS AND CONCLUSION

5.1 Results and Discussion

The right and left gait data are generally similar for people without any deformities or abnormalities in their gait pattern. The data collected are the Accelerometer 3-axis data and the Gyroscope 3-axis data from both the left and the right ankles, i.e., AccelerometerXL, AccelerometerYL, AccelerometerZL, GyroscopeXL, GyroscopeYL, GyroscopeZL, AccelerometerXR, AccelerometerYR, AccelerometerZR, GyroscopeXR, GyroscopeYR, GyroscopeZR. K.C. Lan and W.Y. Shih [24] interpreted the measurements of vertical accelerations during the walking. They mentioned three events which occur during walking-heel-off ground, stance, and heel touching ground. Based on their study, we tried to represent out data by plotting a single trial of data of Person1, and tried to identify a step, heel off ground, and heel touching the ground motions on them. Figure 11 and Figure 12, below show these illustrations for the left and the right sensor data.

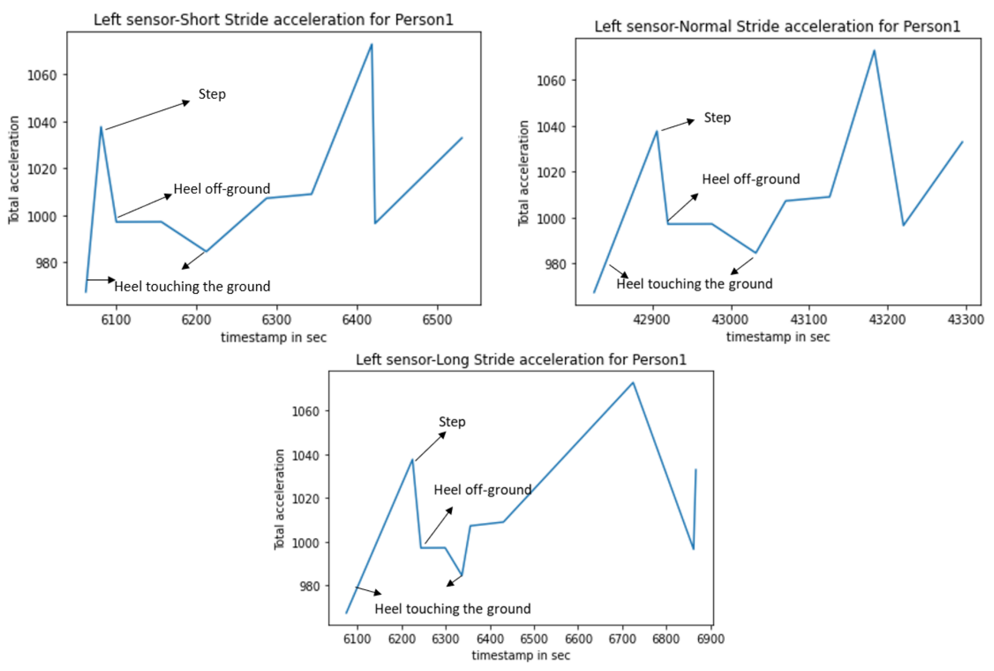


Figure 11. Left sensor data illustration for Person1

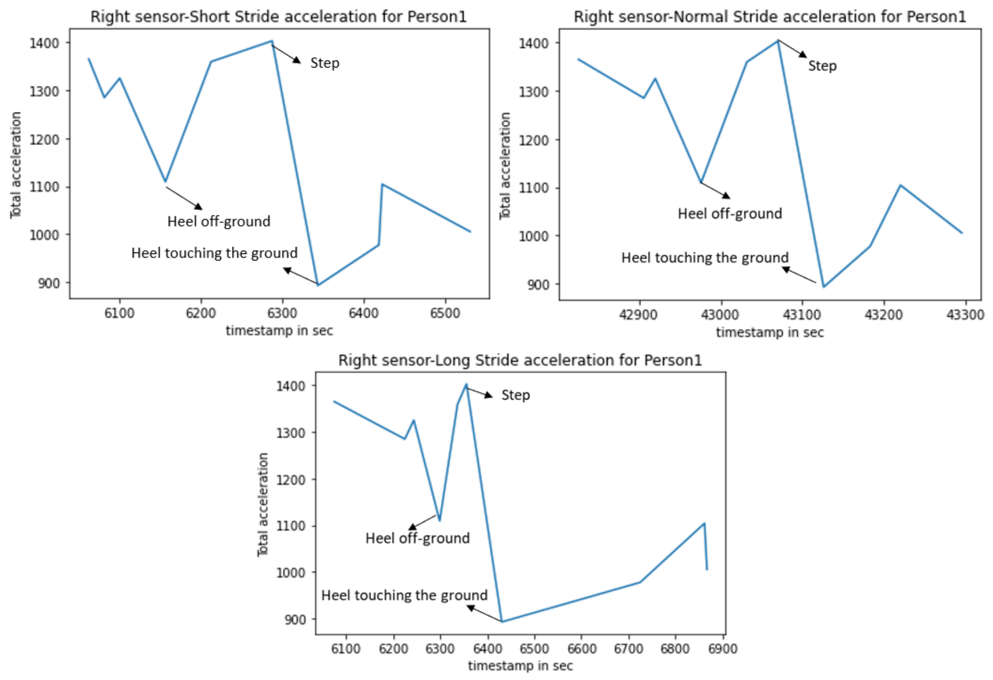


Figure 12. Right sensor data illustration for Person1

Three machine learning algorithms are used to evaluate the performance, namely Logistic Regression, Convolution Neural Network (CNN), and Long Short Term Memory (LSTM). Feature selection was also carried out for CNN based on the correlation values with other features. Two features with highest correlation with the other features were eliminated. Using the raw data, without any preprocessing, gives low results, as it includes a lot of noise. Table 1 shows the accuracy results using the raw data, without any preprocessing.

Models	**Logistic Regression**	**CNN without feature selection**	**CNN with feature selection**	**LSTM**
Accuracy	28.38%	68.57%	68.35%	61.52%

Table 1. Accuracy results for raw data

For further preprocessing, the left and right accelerometer and gyroscope data are combining by taking their absolute mean values at each instance. Once this is done, we calculate the overall mean for each of the six features- AccX, AccY, AccZ, GyroX, GyroY, GyroZ for short, normal, and long strides. Table 2, Table 3, and Table 4 show the mean values for each of the six features, rounded to two decimal places, for short, normal, and long strides respectively.

Mean of AccX	**Mean of AccY**	**Mean of AccZ**	**Mean of GyroX**	**Mean of GyroY**	**Mean of GyroZ**
1023.06	199.31	252.35	36.35	61.70	21.40

Table 2. Mean Values for Short Stride

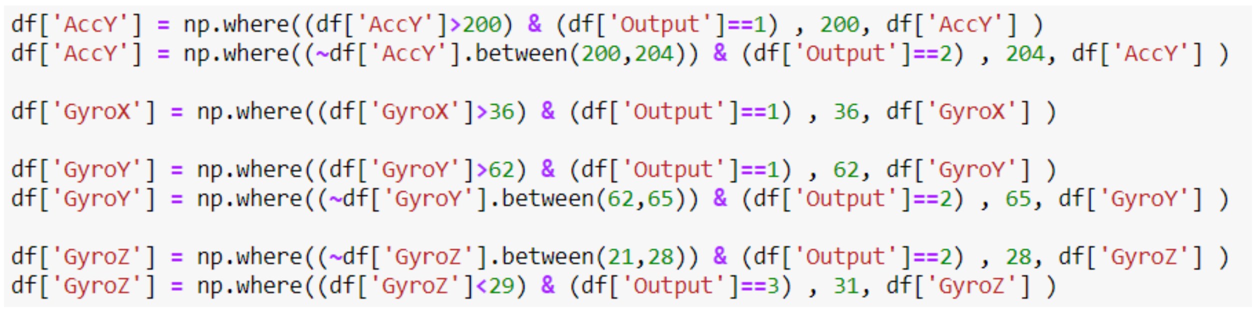
Mean of AccX	**Mean of AccY**	**Mean of AccZ**	**Mean of GyroX**	**Mean of GyroY**	**Mean of GyroZ**
1046.81	203.97	243.13	66.86	64.81	28.25

Table 3. Mean Values for Normal Stride

Mean of AccX	**Mean of AccY**	**Mean of AccZ**	**Mean of GyroX**	**Mean of GyroY**	**Mean of GyroZ**
1062.12	231.47	328.67	56.60	86.81	31.05

Table 4. Mean Values for Long Stride

Classifying among the three strides can be a little complex as the motion is the same for the three strides, i.e., walking. This can result in certain values being close to each other, thus causing errors while classifying. As can be seen from Table 2, Table 3, and Table 4, the mean values are very close for AccY and GyroY in case of short and normal strides, and GyroZ in case of normal and long strides. Also, there is a clear difference in the GyroX value in case of short stride. Taking these into consideration, we set certain conditions as shown in Figure 11, to improve the model performance. We convert certain values, which are very noisy, to their mean values according to the set condition.



```
df['AccY'] = np.where((df['AccY']>200) & (df['Output']==1) , 200, df['AccY'] )
df['AccY'] = np.where((~df['AccY'].between(200,204)) & (df['Output']==2) , 204, df['AccY'] )

df['GyroX'] = np.where((df['GyroX']>36) & (df['Output']==1) , 36, df['GyroX'] )

df['GyroY'] = np.where((df['GyroY']>62) & (df['Output']==1) , 62, df['GyroY'] )
df['GyroY'] = np.where((~df['GyroY'].between(62,65)) & (df['Output']==2) , 65, df['GyroY'] )

df['GyroZ'] = np.where((~df['GyroZ'].between(21,28)) & (df['Output']==2) , 28, df['GyroZ'] )
df['GyroZ'] = np.where((df['GyroZ']<29) & (df['Output']==3) , 31, df['GyroZ'] )
```

Figure 13. Conditions to convert noisy values to their mean

Once these preprocessing steps are carried out, we use the three machine learning algorithms, to once again carry out the training process and compare their accuracy values with each other and with the previous accuracy values obtained for the raw data. The accuracy values for the preprocessed data are shown in Table 5.

Models	**Logistic Regression**	**CNN**	**LSTM**
Accuracy	65.52%	91.43%	97.71%

Table 5. Accuracy values of the preprocessed data

As we can see from Table 1 and Table 5, the accuracy has significantly increased in the latter from the former for all three models. Among the three, CNN and LSTM have performed exceptionally well at 91.43% and 97.71% accuracies respectively. Figure 12 and Figure 13, show the accuracy and loss graphs for test and train data for CNN and LSTM respectively.

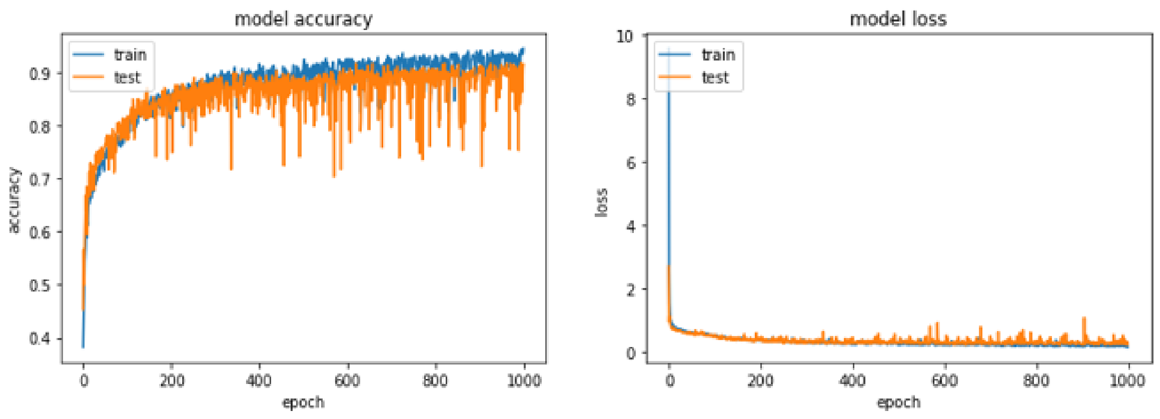


Figure 14. Accuracy and Loss Graphs for CNN

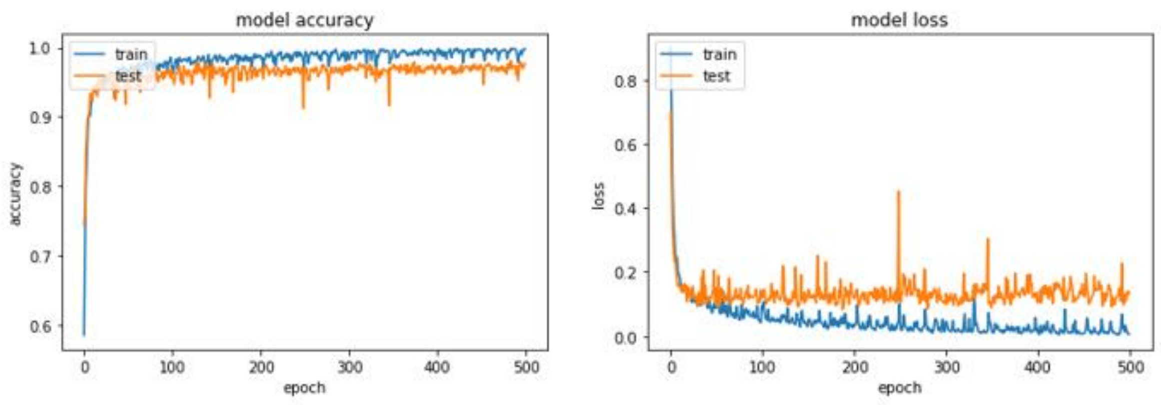


Figure 15. Accuracy and Loss Graphs for LSTM

Another method of preprocessing using Kalman Filter was used and the accuracies were obtained. Kalman Filter is one of the most important algorithms used to estimate hidden variables based on the inaccuracies and uncertainties in measurements. In this method, all the features from the raw data are used and this data is passed through the Kalman Filter. Figure 16, shows the sample of data before and after using Kalman Filter.

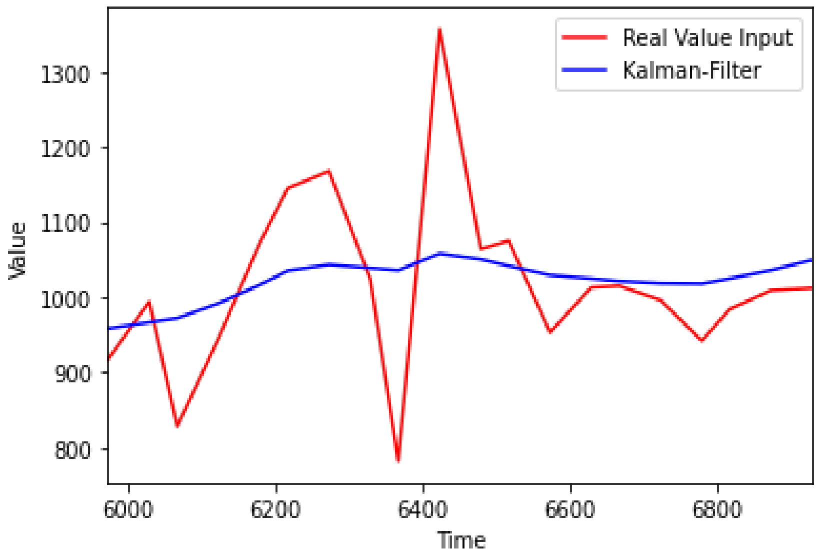


Figure 16. Sample of data before and after using Kalman Filter

Two machine learning algorithms- CNN and LSTM, are used and the accuracies are calculated. After using Kalman Filter, there was a significant improvement in the performance of the two models. CNN gave an accuracy of 91.24% and LSTM gave an accuracy of 91.62%.

Figure 14, Figure 15, and Figure 16, show the snapshots of the data from five people captured by the depth sense cameras along with the Skeleton Tracking SDK for short stride, normal stride and long stride respectively.

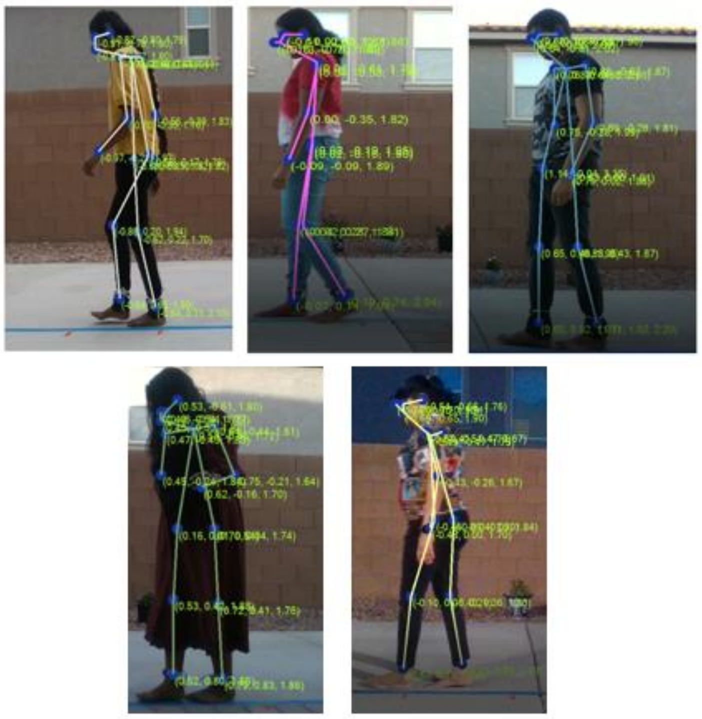


Figure 17. Snapshots captured by D435i with Skeleton Tracking SDK for short stride

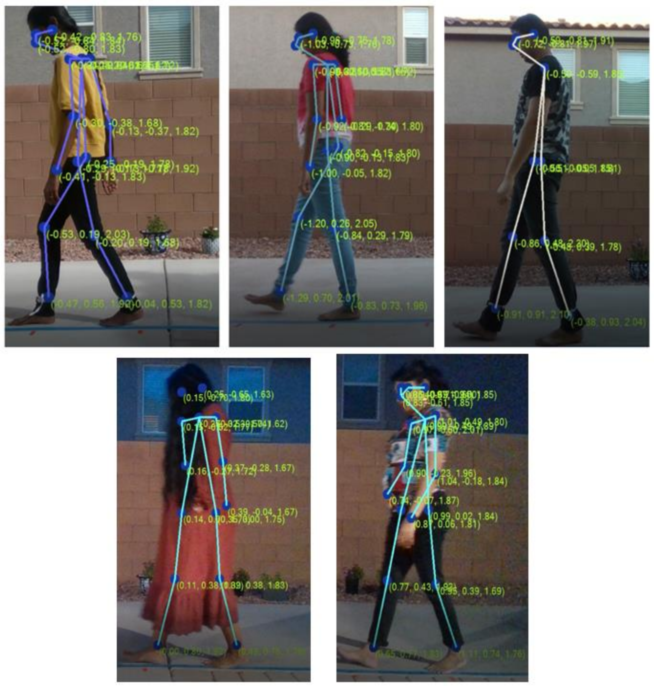


Figure 18. Snapshots captured by D435i with Skeleton Tracking SDK for normal stride

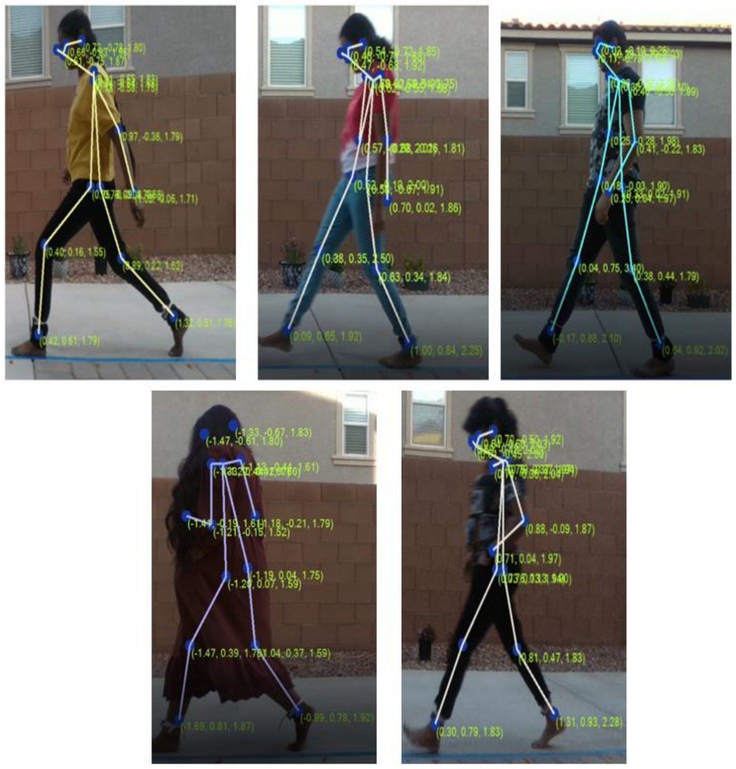


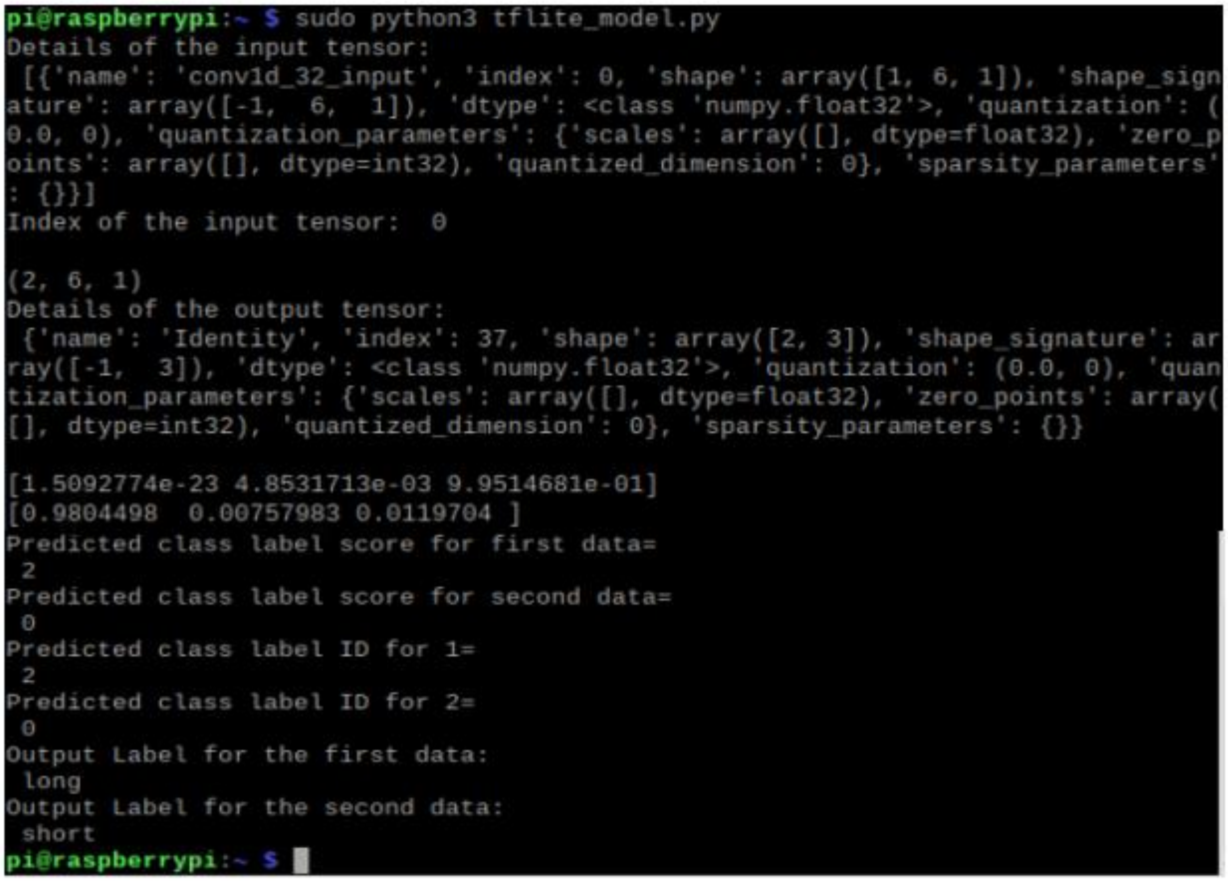
Figure 19. Snapshots captured by D435i with Skeleton Tracking SDK for long stride

The Skeleton Tracking SDK provides the 3D coordinates in real time. The approximate step lengths (in meters) for the five people obtained from the images are given in Table 6.

	Person 1	**Person 2**	**Person 3**	**Person 4**	**Person 5**
Short Stride	0.242	0.265	0.524	0.272	0.311
Normal Stride	0.442	0.464	0.534	0.502	0.466
Long Stride	0.906	0.986	0.815	0.702	1.115

Table 6. The approximate step lengths (in meters) for the five people

After deploying the TFLite model onto the Raspberry Pi 4 and by running all the inferences and interpretations, the output is displayed as shown in Figure 17. There are two input values given to the model and the model makes the predictions as long and short respectively.

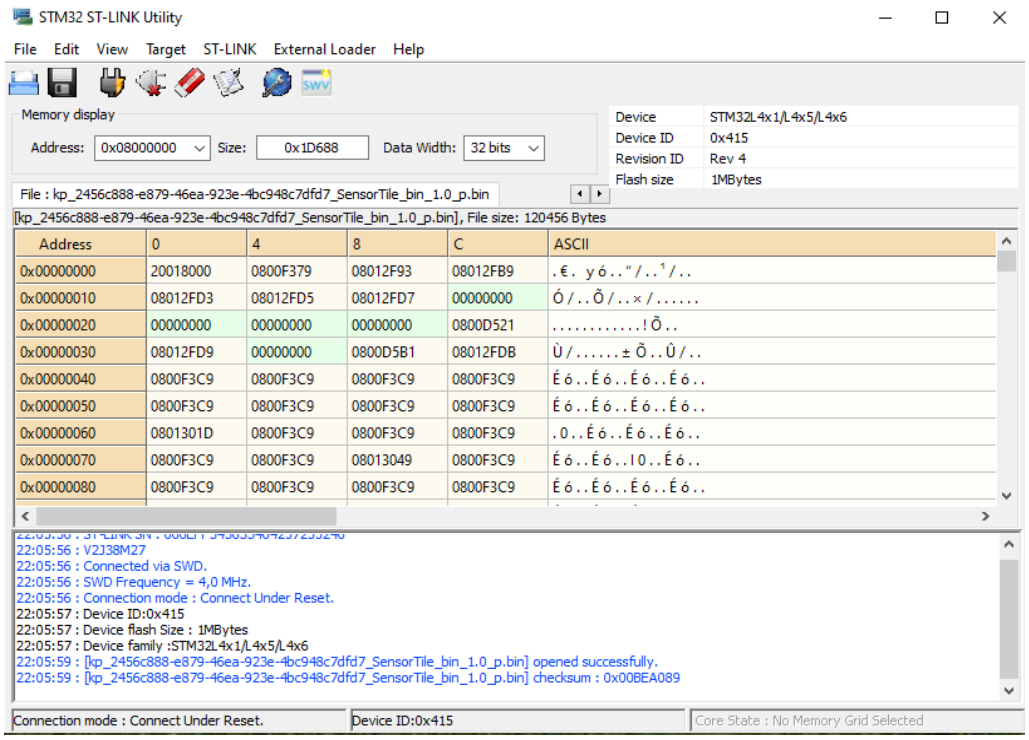


```
pi@raspberrypi:~ $ sudo python3 tflite_model.py
Details of the input tensor:
 [{'name': 'conv1d_32_input', 'index': 0, 'shape': array([1, 6, 1]), 'shape_sign
ature': array([-1,  6,  1]), 'dtype': <class 'numpy.float32'>, 'quantization': (
0.0, 0), 'quantization_parameters': {'scales': array([], dtype=float32), 'zero_p
oints': array([], dtype=int32), 'quantized_dimension': 0}, 'sparsity_parameters'
: {}}]
Index of the input tensor:  0

(2, 6, 1)
Details of the output tensor:
 {'name': 'Identity', 'index': 37, 'shape': array([2, 3]), 'shape_signature': ar
ray([-1,  3]), 'dtype': <class 'numpy.float32'>, 'quantization': (0.0, 0), 'quan
tization_parameters': {'scales': array([], dtype=float32), 'zero_points': array(
[], dtype=int32), 'quantized_dimension': 0}, 'sparsity_parameters': {}}

[1.5092774e-23 4.8531713e-03 9.9514681e-01]
[0.9804498  0.00757983 0.0119704 ]
Predicted class label score for first data=
 2
Predicted class label score for second data=
 0
Predicted class label ID for 1=
 2
Predicted class label ID for 2=
 0
Output Label for the first data:
 long
Output Label for the second data:
 short
pi@raspberrypi:~ $ █
```

Figure 20. Output of the ML model on Raspberry Pi 4

The model was also deployed onto the two SensorTile devices for future use. To achieve this, we have used the SensiML TinyML framework. This framework helps in creating Machine Learning models compatible with small, low-powered devices such as sensors. There are two main toolkits SensiML offers: Data Capture Lab, and Analytics Studio. Using Data Capture Lab one can either capture live data or upload a CSV file of already captured data. This data, after labelling, is used by the Analytics Studio to create the Machine Learning model. After carrying out the testing and validation, we download the Knowledge Pack which contains the binary file, which can be flashed on the SensorTile device.

SensiML also contains firmware to collect data on SD card using Data Capture Lab. The Knowledge Pack created can then be flashed onto the device using STM32 ST-Link Utility application. Figure 21, shows the Knowledge Pack binary file flashed onto the SensorTile device. Once this is completed, the Knowledge Pack recognition results can be viewed by plugging in the SensorTile device via a serial USB cable, connecting it to a terminal emulator such as Tera Term, and setting the serial speed in the terminal emulator to 115200.



Address: 0x08000000 Size: 0x1D688 Data Width: 32 bits

Device	STM32L4x1/L4x5/L4x6
Device ID	0x415
Revision ID	Rev 4
Flash size	1MBytes

File : kp_2456c888-e879-46ea-923e-4bc948c7dfd7_SensorTile_bin_1.0_p.bin

[kp_2456c888-e879-46ea-923e-4bc948c7dfd7_SensorTile_bin_1.0_p.bin], File size: 120456 Bytes

Address	0	4	8	C	ASCII
0x00000000	20018000	0800F379	08012F93	08012FB9	.€. yó..“/..¹/..
0x00000010	08012FD3	08012FD5	08012FD7	00000000	Ó/..Õ/..×/......
0x00000020	00000000	00000000	00000000	0800D521	............!Õ..
0x00000030	08012FD9	00000000	0800D5B1	08012FDB	Ù/......±Õ..Û/..
0x00000040	0800F3C9	0800F3C9	0800F3C9	0800F3C9	Éó..Éó..Éó..Éó..
0x00000050	0800F3C9	0800F3C9	0800F3C9	0800F3C9	Éó..Éó..Éó..Éó..
0x00000060	0801301D	0800F3C9	0800F3C9	0800F3C9	.0..Éó..Éó..Éó..
0x00000070	0800F3C9	0800F3C9	08013049	0800F3C9	Éó..Éó..I0..Éó..
0x00000080	0800F3C9	0800F3C9	0800F3C9	0800F3C9	Éó..Éó..Éó..Éó..

```
22:05:56 : V2J38M27
22:05:56 : Connected via SWD.
22:05:56 : SWD Frequency = 4,0 MHz.
22:05:56 : Connection mode : Connect Under Reset.
22:05:57 : Device ID:0x415
22:05:57 : Device flash Size : 1MBytes
22:05:57 : Device family :STM32L4x1/L4x5/L4x6
22:05:59 : [kp_2456c888-e879-46ea-923e-4bc948c7dfd7_SensorTile_bin_1.0_p.bin] opened successfully.
22:05:59 : [kp_2456c888-e879-46ea-923e-4bc948c7dfd7_SensorTile_bin_1.0_p.bin] checksum : 0x00BEA089
```

Connection mode : Connect Under Reset. | Device ID:0x415 | Core State : No Memory Grid Selected

Figure 21. Knowledge Pack Binary File flashed onto the SensorTile Device

5.2 Conclusion

TinyML is an emerging technology and is very useful in wide variety of applications. It can help in eradicating the issues faced by the IoT devices. In this study, we make use of TinyML, which combines embedded systems and machine learning techniques, to classify between three types of strides, namely short, normal, and long strides. We have made use of two IMU sensors and a raspberry pi 4 device to carry out the collection of data and for final analysis. Three types of machine algorithms are used with preprocessing techniques on the raw data. Ground marking along with depth sense camera and skeleton tracking SDK provide additional details and help to establish the ground truth.

From this study, we can conclude that machine learning techniques along with the embedded system can be a powerful system for gait classification and analysis. Among the machine learning algorithms used, we could see that neural networks performed exceptionally well as compared to regression algorithm, when non-linearities are involved. Also, preprocessing of data is a very important step, which helps in reducing the noise in the data and improve performance. The depth sense camera used with the skeleton tracking SDK help in analyzing and calculating the stride lengths.

APPENDIX A: IMPORTANT SNIPPETS OF CODE

1. **Preprocessing data to convert the noisy data into their mean value**

```
df['AccY'] = np.where((df['AccY']>200) & (df['Output']==1) , 200, df['AccY'] )

df['AccY'] = np.where((~df['AccY'].between(200,204)) & (df['Output']==2) , 204
df['AccY'] )

df['GyroX'] = np.where((df['GyroX']>36) & (df['Output']==1) , 36, df['GyroX'] )

df['GyroY'] = np.where((df['GyroY']>62) & (df['Output']==1) , 62, df['GyroY'] )
df['GyroY'] = np.where((~df['GyroY'].between(62,65)) & (df['Output']==2) , 65,
df['GyroY'] )

df['GyroZ'] = np.where((~df['GyroZ'].between(21,28)) & (df['Output']==2) , 28,
df['GyroZ'] )
df['GyroZ'] = np.where((df['GyroZ']<29) & (df['Output']==3) , 31, df['GyroZ'] )
```

2. **Logistic Regression Model**

```
lr_model = LogisticRegression(solver='liblinear', random_state=0, C=0.7)
lr_model.fit(X_train, y_train.ravel())
lr_predict_test = lr_model.predict(X_test)
```

3. **CNN Model**

```
model = keras.models.Sequential()
model.add(Conv1D(32, 2, activation="relu", input_shape=(6,1)))
model.add(Dense(16, activation="relu"))
model.add(MaxPooling1D())
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(100, activation='relu'))
model.add(Dense(3, activation = 'softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer = "adam", metrics =
['accuracy'])
```

4. LSTM Model

```
model = Sequential()
model.add(LSTM(200, input_shape=(12,1)))
model.add(Dropout(0.5))
model.add(Dense(100, activation='relu'))
model.add(Dense(3,activation = 'softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer = "adam", metrics =
['accuracy'])
```

5. Capturing data from two SensorTile devices on raspberry pi 4 via Bluetooth (BLE)

*Please note: The main libraries and code for importing the manager and carrying out the discovery process is obtained from the below link: https://github.com/STMicroelectronics/BlueSTSDK_Python

```
Pseudocode:
1)  Set number of notifications
2)  Carry out the discovery process
3)  Call the feature listener
4)  Define a function to take the input and store them in a text file
5)  Start discovery
            If devices are found:
                    Select the devices to be connected and connect
            Else:
                    Disconnect
6)  Listen to features of both the devices
            Select features from both devices  #Accelerometer and Gyroscope
7)  Enable notifications for both devices
8)  Wait for notification from both
9)  Call the function to store data into a text file
10) Disable notifications once data is collected
```

6. **Saving the TensorFlow model and convert it into a TFLite Model**

```
# save model and architecture to single file
model.save("model.h5")
print("Saved model to disk")

#Convert to a TFLite model and save it
converter = tf.compat.v1.lite.TFLiteConverter.from_keras_model_file('model.h5')
tflite_model = converter.convert()
```

7. **Running Inferences on the embedded device after the TFlite is deployed to make predictions**

```
import tflite_runtime
from tflite_runtime.interpreter import Interpreter
import numpy as np

model_path = "/home/pi/Desktop/TF_Lite/model.tflite"
interpreter = Interpreter(model_path)

print("Details of the input tensor:\n", interpreter.get_input_details())

tensor_index =  interpreter.get_input_details()[0]['index']
print("Index of the input tensor: ", tensor_index, end="\n\n")

#Supply input. If its in some other form data has to be converted into the format
accepted by the model. Here it’s shown for just two input data
batch =
np.array([[[1062.0],[235.7],[350.6],[57.0],[88.34],[32.56]],[[1023],[200],[252],[35],[62],[
20]]], dtype=np.float32)
print(batch.shape)

input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

interpreter.resize_tensor_input(input_details[0]['index'], (2,6,1))
```

```
 interpreter.resize_tensor_input(output_details[0]['index'], (2,5))

 interpreter.allocate_tensors()

 input_tensor=batch
 interpreter.set_tensor(input_details[0]['index'], batch)

 interpreter.invoke()

 output_details = interpreter.get_output_details()[0]
 print("Details of the output tensor:\n", output_details, end="\n\n")

 scores1 = interpreter.get_tensor(output_details['index'])[0]
 scores2 = interpreter.get_tensor(output_details['index'])[1]
 print(scores1)
 print(scores2)

print("Predicted class label score for first data=\n", np.argmax(scores1))
print("Predicted class label score for second data=\n", np.argmax(scores2))

score_pred1 = np.argmax(scores1)
score_pred2 = np.argmax(scores2)
print("Predicted class label ID for 1=\n", score_pred1)
print("Predicted class label ID for 2=\n", score_pred2)

label_path = "/home/pi/Desktop/TF_Lite/label.txt"

with open(label_path, 'r') as f:
   labels = [line.strip() for i, line in enumerate(f.readlines())]

label_id1 = score_pred1
label_id2 = score_pred2

#Return the classification label of the image
classification_label1 = labels[label_id1]
print("Output Label for the first data:\n", classification_label1)
classification_label2 = labels[label_id2]
print("Output Label for the second data:\n", classification_label2)
```

APPENDIX B: DATA AND VIDEOS CAPTURED BY DEPTH SENSE CAMERA

The data has been loaded on to Google Drive. To view the data and videos, please use the below link:

https://drive.google.com/drive/folders/12BLaVkQ8kMsbVhhEktHH3ChXOkB0cifd?usp=sharing

In the link, there are three data files uploaded-Finaldata_withTS.csv, Finaldata.csv and the MeanFinal.csv. The first two data files are the raw data with and without timestamp respectively, and the final file is the data obtained by taking the average/mean of the left and right sensor data. The Stride Data-Videos has the data captured by the Intel RealSense depth camera D435i with the skeleton tracking algorithm by Cubemos integrated with it. There are videos captured from the front and the side views for all three types of strides-short, normal and long.

REFERENCES

[1] Loudon J, et al. The clinical orthopedic assessment guide. 2nd ed. Kansas: Human Kinetics, 2008. p.395-408.

[2] Abdul Saboot, et al., “Latest Research Trends in Gait Analysis Using Wearable sensors and Machine Learning: A systematic review”, IEEE, Volume 8, September 2020

[3] “The Gait Cycle: Phases, Parameters to Evaluate & Technology” Tekscan, https://www.tekscan.com/blog/medical/gait-cycle-phases-parameters-evaluate-technology

[4] Mohammad Omar Derawi, “Accelerometer-Based Gait Analysis, A survey”, ResearchGate, January 2010

[5] “How TinyML can transform IoT applications across industries”, TechAhead, October 2020, https://www.techaheadcorp.com/blog/tinyml-transform-iot-applications/

[6] Wesley Niswander and Kimberly Kontson, “Evaluating the Impact of IMU Sensor Location and Walking Task on Accuracy of Gait Event Detection Algorithms”, MDPI, June 2021

[7] Cecilia Monoli, et al., “Land and Underwater Gait Analysis Using Wearable IMU”, IEEE Sensors Journal, Vol.21, No. 9, May 2021

[8] Kirk Patrick Rustia , Shayan Hosseinpouli, “STMicroelectronics SensorTile Reference Design: Wearable Gait Classification Prototype using Stride Measurements”, UCLA

[9] Zexia He, Tao Liu, Jingang Yi, “A Wearable Sensing and Training System: Towards Gait Rehabilitation for Elderly Patients with Knee Osteoarthritis”, IEEE Sensors Journal, Volume 19, Issue 4, July 2019

[10] Xin Liu et al., “Wearable Devices for Gait Analysis in Intelligent Healthcare”, Frontiers in Computer Science, May 2021

[11] Julianna Delua, SME, IBM Analytics, Data Science/Machine Learning, “Supervised vs. Unsupervised Learning: What’s the Difference?” https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning

[12] Shweta Bhatt, “Reinforcement Learning 101”, towards data science, Mar 2018, https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

[13] Manav Mandal, “Introduction to Convolutional Neural Networks (CNN)”, May 2021, https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

[14] Niklas Donges, “A Guide to RNN: Understanding Recurrent Neural Networks and LSTM Networks”, July 2021, https://builtin.com/data-science/recurrent-neural-networks-and-lstm

[15] Pooja Gupta, “Components of Embedded System”, Educba, https://www.educba.com/components-of-embedded-system/

[16] Sérgio Branco, André G. Ferreira, Jorge Cabral, “Machine Learning in Resource-Scarce Embedded Systems, FPGAs, and End-Devices: A Survey”, MDPI, November 2019

[17] Taiwo Samuel Ajani, Agbotiname Lucky Imoize, Aderemi A. Atayero, "An Overview of Machine Learning within Embedded and Mobile Devices–Optimizations and Applications", Sensors (Basel), June 2021

[18] C. Banbury, C. Zhou, I. Fedorov, R. Matas, U. Thakker, D. Gope, V. Janapa Reddi, M. Mattina, P. Whatmough, "MicroNets: Neural network architectures for deploying TinyML Applications on Commodity Microcontrollers", Proceedings of the 4th MLSys Conference; San Jose, CA, USA. 4–7 April 2021

[19] Mohammad Ali Arabi, Viktoria Schwarz, "General Constraints in Embedded Machine Learning and How to Overcome Them — A Survey Paper", ResearchGate, July 2019

[20] STEVAL-STLKT01V1, Data brief- SensorTile development kit, STMicroelectronics documentation.

[21] Raspberry Pi 4 Computer Model B datasheet, January 2021, https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-product-brief.pdf

[22] Beginner's Guide to Depth, intel RealSense, July 2019, https://www.intelrealsense.com/beginners-guide-to-depth/

[23] Pete Warden, Daniel Situnayake, ed. (2019), TinyML Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers, First edition. O'reilly Media, CA

[24] Kun-Chan Lan, Wen-Yuah Shih, "Using Simple Harmonic Motion to Estimate Walking Distance for Waist-mounted PDR", 2012 IEEE Wireless Communications and Networking Conference (WCNC), April 2012

CURRICULUM VITAE

Graduate College

University of Nevada, Las Vegas

Priyanka Rajendra

priyanka.rajendra95@gmail.com

Degrees:

Bachelor of Engineering (B.E.), Electronics and Communication, 2016

Visvesvaraya Technological University, Belgaum, India

Thesis Title: TinyML for Gait Stride Classification

Thesis Examination Committee:

Advisory Committee Chair, Dr. Venkatesan Muthukumar

Advisory Committee Member, Dr. Mei Yang

Advisory Committee Member, Dr. Emma Regentova

Advisory Committee Member, Dr. Si Jung Kim

Graduate Faculty Representative, Dr. Juyeon Jo